UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Programación II – Curso 2021/22

Práctica 2: Enunciado

1. El Problema

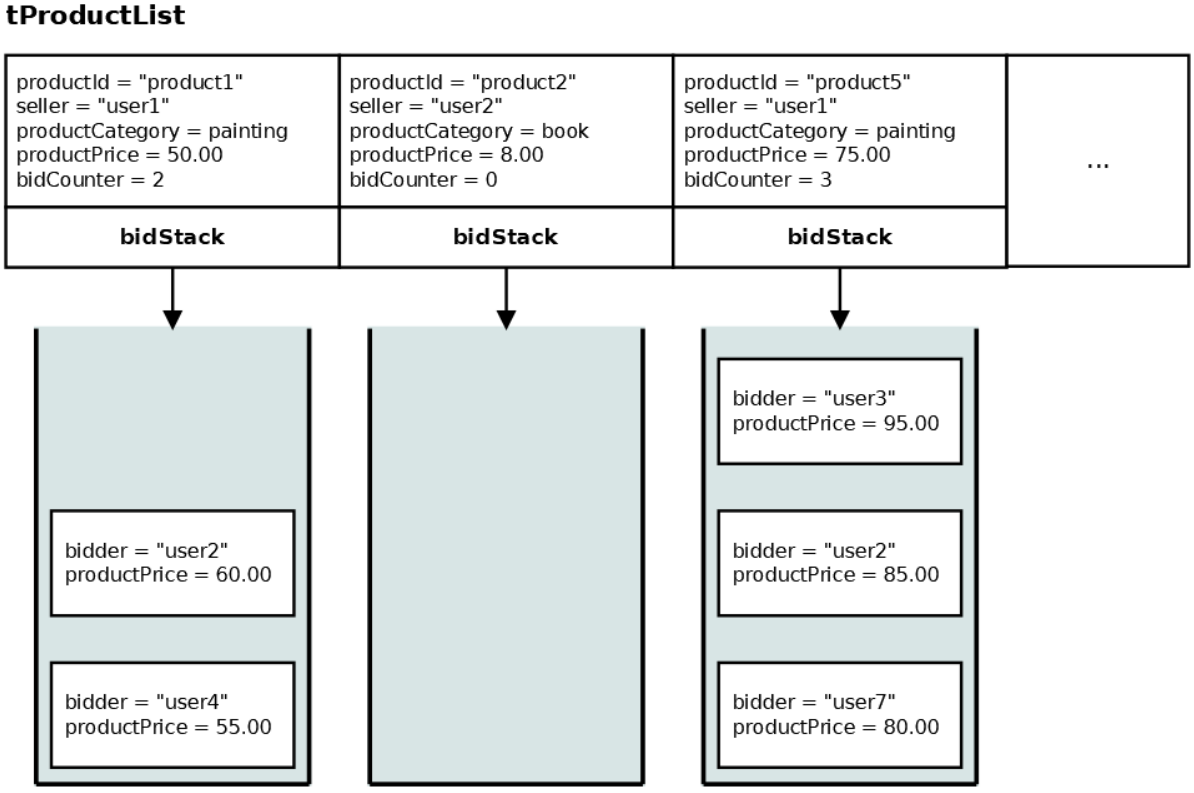
La segunda práctica consistirá en añadir un conjunto de nuevas funcionalidades al programa BIDFIC desarrollado en la primera práctica. En concreto, se incluirá una **pila de pujas de precios para cada producto**. El objetivo de esta práctica es comprender el funcionamiento e implementar varios tipos abstractos de datos (TADs) así como manejar interdependencias entre ellos.

2. Estructuras de Datos

Para resolver este problema se utilizarán 2 tipos abstractos de datos (TADs):

- Una Lista Ordenada (TAD ProductList) para almacenar la lista de productos.
- Una Pila (TAD BidStack) para almacenar las pujas por cada producto en la subasta.

Estos dos TADs se unen para permitir que cada producto de BIDFIC, esto es, cada nodo de la lista ordenada, tenga su propia pila de pujas, tal y como representa la siguiente figura



2.1 Librería `types.h`

Algunos tipos de datos comunes se definirán en el fichero `types.h`, ya que se utilizarán en varios TADs.

`NAME_LENGTH_LIMIT`	Longitud máxima de userId y productId (constante)
`tUserId`	Identificador de un usuario (`string`)
`tProductId`	Identificador de un producto (`string`)
`tProductCategory`	Categoría de producto (tipo enumerado: `{book, painting}`)
`tProductPrice`	Precio del producto (`float`)
`tBidCounter`	Contador de pujas (`int`)

2.2 TAD Product List

Para mantener la lista de productos el sistema utilizará un TAD Lista ordenada con implementación **dinámica, simplemente enlazada** (`product_list.c` y `product_list.h`).

2.2.1. Tipos de datos incluidos en el TAD Product List

`tList`	Representa una lista de productos **ordenada alfabéticamente** por el identificador de producto (`productId`)
`tItemL`	Datos de un elemento de la lista (un producto). Compuesto por: • `seller` de tipo `tUserId` • `productId` de tipo `tProductId` • `productCategory` de tipo `tProductCategory` • `productPrice` de tipo `tProductPrice` • `bidCounter` de tipo `tBidCounter` • `bidStack`: de tipo `tStack` (pila con las pujas recibidas por el producto)
`tPosL`	Posición de un elemento de la lista de productos
`LNULL`	Constante usada para indicar posiciones nulas de la lista

2.2.2. Operaciones incluidas en el TAD Product List

Las operaciones del TAD Lista ordenada son idénticas a las indicadas en la Práctica 1 (consulte el enunciado de dicha práctica), únicamente cambian de especificación las siguientes operaciones:

- `insertItem (tItemL, tList) → tList, bool`
  Inserta un elemento en la Lista de forma **ordenada** por el campo `productId`. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.

PostCD: Las posiciones de los elementos de la lista posteriores a la del elemento insertado pueden haber variado.

- `deleteAtPosition (tPosL, tList) → tList`
  Elimina de la lista el elemento que ocupa la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista **y el producto en dicha posición tiene una pila de pujas vacía.**
  PostCD: Las posiciones de los elementos de la lista posteriores a la de la posición eliminada pueden haber variado.

Asimismo, la operación `findItem` no cambia su especificación, pero en su implementación debería tener en cuenta que la lista está ordenada.

2.3 TAD Bid Stack

Este TAD permitirá almacenar las pujas recibidas por un producto, y su implementación corresponde a una **pila estática** para un máximo de 25 pujas (`bid_stack.c` y `bid_stack.h`).

2.3.1. Tipos de datos incluidos en el TAD Bid Stack

`tStack`	Representa una pila de pujas
`tItemS`	Datos de un elemento de la pila, es decir, una puja. Compuesto por: • `bidder`: de tipo `tUserId` (el usuario que hace la puja) • `productPrice`: de tipo `tProductPrice` (el importe que ofrece)
`tPosS`	Posición de un elemento de la pila de pujas
`SNULL`	Constante usada para indicar posiciones nulas de la pila

2.3.2. Operaciones incluidas en el TAD Bid Stack

Una precondición común para todas estas operaciones (salvo `createEmptyStack`) es que la pila debe estar previamente inicializada:

- `createEmptyStack (tStack) → tStack`
  Crea una pila vacía.
  PostCD: la pila no tiene elementos.

- `push (tItemS, tStack) → tStack, bool`
  Inserta un elemento en la cima de la pila. **Devuelve un valor `true` si el elemento fue apilado; `false` en caso contrario.**

- `pop (tStack) → tStack`
  Elimina de la pila el elemento situado en la cima.
  PreCD: La pila no está vacía.

- `peek (tStack) → tItemS`
  Recupera el elemento de la cima de la pila sin eliminarlo.
  PreCD: La pila no está vacía.

- `isEmptyStack (tStack) → bool`
  Determina si una pila está vacía o no.

3. Descripción de la tarea

La tarea consiste en implementar un único programa principal (`main.c`) que haga uso de la **Product List** y de **Bid Stack** y que procese las peticiones sobre los productos de BIDFIC con el siguiente formato:

`N productId userId productCategory productPrice`	***[N]ew**:* Alta de un nuevo producto
`D productId`	***[D]elete***: Baja de un producto
`B productId userId productPrice`	***[B]id***: Puja por un determinado producto
`A productId`	***[A]ward:*** Se asigna el ganador de una puja
`W productId userId`	***[W]ithdraw***: La máxima puja actual del producto es retirada
`R`	***[R]emove:*** Elimina los productos sin pujas
`S`	***[S]tats:*** Listado de los productos actuales de BIDFIC y sus datos

En el programa principal se implementará un bucle que procese una a una las peticiones sobre los productos. Para simplificar el desarrollo y las pruebas, el programa no necesitará introducir ningún dato por teclado, sino que leerá y procesará las peticiones de productos contenidas en un fichero (ver documento `EjecucionScript.pdf`). En cada iteración del bucle, el programa leerá del fichero una nueva petición y la procesará. Para facilitar la corrección de la práctica las peticiones del fichero van numeradas correlativamente.

Para cada línea del fichero de entrada, el programa:

1. Muestra una cabecera con la operación a realizar. Esta cabecera está formada por una primera línea con 20 asteriscos y una segunda línea que indica la operación tal y como se muestra a continuación:

```
********************
NN␣T:␣product␣PP␣seller/bidder␣UU␣category␣CC␣price␣ZZ
```

donde `NN` es el número de petición, `T` es el tipo de operación (`N`, `D`, `B`, `A`, `W`, `R` o `S`); `PP` es el identificador del producto; `UU` es identificador del usuario (precedido de la palabra `seller` o `bidder` dependiendo de la operación); `CC` es la categoría del producto; `ZZ` es el precio de este (con dos decimales); y `␣` indica un espacio en blanco. Sólo se imprimirán los parámetros necesarios; es decir, para una petición ***[S]tats*** se mostrará únicamente "`01 S`", mientras que para una petición ***[N]ew*** se mostrará "`01 N: product Product1 seller User2 category book price 15.00`".

2. Procesa la petición correspondiente:

- Si la operación es ***[N]ew***, se añadirá el producto a la lista de productos, con su correspondiente identificador de producto, el identificador de usuario de su vendedor, su categoría y su precio. El contador de pujas del producto se inicializará a 0 y se le asociará una nueva pila de pujas vacía. Además, se imprimirá el mensaje:

  ```
  * New: product PP seller UU category CC price ZZ
  ```

  Si ya existiese un producto con ese identificador o el producto no se pudiera insertar, se imprimirá el siguiente mensaje:

  ```
  + Error: New not possible
  ```

- Si la operación es ***[D]elete***, se buscará el producto en la lista, se borrará de la lista, **eliminando todas las pujas existentes**, y se imprimirá el siguiente mensaje:

  ```
  * Delete: product PP seller UU category CC price ZZ bids II
  ```

  Si no existiese ningún producto con ese identificador, se imprimirá el siguiente mensaje:

  ```
  + Error: Delete not possible
  ```

- Si la operación es ***[B]id***, se buscará el producto y, si la nueva puja supera la puja más alta actual (o el precio original, si no hubiese pujas), entonces: (i) se añadirá la nueva puja a su pila; (ii) se actualizará el contador de pujas; y (iii) se mostrará el siguiente mensaje con el precio de la puja:

  ```
  * Bid: product PP bidder UU category CC price ZZ bids II
  ```

  Si no existiese ningún producto con dicho identificador (`productId`), el pujador (`bidder`) y el vendedor (`seller`) fuesen la misma persona, el precio de la puja no fuese superior a la puja más alta actual, o la pila estuviese llena, se imprimirá entonces el mensaje:

  ```
  + Error: Bid not possible
  ```

- Si la operación es ***[A]ward***, se buscará el producto y se mostrará el siguiente mensaje:

  ```
  * Award: product PP bidder UU category CC price ZZ
  ```

  donde `UU` es el ganador de la puja y `ZZ` es el precio final. Dado que el producto ha sido vendido, deberá ser eliminado de la lista de productos tras vaciar su pila de pujas.

  Si no existiese ningún producto con dicho identificador (`productId`) o su pila de pujas estuviese vacía, se imprimirá el mensaje:

```
+ Error: Award not possible
```

En este último caso, con la pila vacía, no se eliminará el producto de la lista.

- Si la operación es ***[W]ithdraw***, se buscará el producto, se eliminará su mejor puja actual, se decrementará el contador de pujas y se mostrará el siguiente mensaje:

```
* Withdraw: product PP bidder UU category CC price ZZ bids II
```

donde, en este caso, `ZZ` es el importe **de la puja a retirar** e `II` es el número de pujas existentes por ese producto **antes** de proceder a dicha retirada.

Si no existiese ningún producto con ese identificador (`productId`), la pila de pujas estuviese vacía o el `userId` no se correspondiese con el del autor de la puja a retirar, en su lugar se imprimirá el mensaje:

```
+ Error: Withdraw not possible
```

- Si la operación es ***[S]tats***, se mostrará la lista completa de productos actuales de la siguiente forma:

```
Product PP1 seller UU1 category CC1 price ZZ1 bids II1 top bidder BB1
Product PP2 seller UU2 category CC2 price ZZ2. No bids
Product PP3 seller UU3 category CC3 price ZZ3 bids II3 top bidder BB3
...
Product PPn seller UUn category CCn price ZZn bids TTn top bidder BBn
```

donde `ZZi` es el precio original. A continuación, se mostrarán, para cada categoría, el número de productos ofertados en esa categoría, la suma de precios originales de esos productos, así como su precio medio; todo ello con el siguiente formato:

```
Category  Products    Price  Average
Book      %8d %8.2f %8.2f
Painting  %8d %8.2f %8.2f
```

Por último, se mostrará el producto para el que el importe de su mejor puja suponga el mayor incremento (en términos porcentuales) sobre su precio original

```
Top bid: Product PP seller UU category CC price ZZ bidder BB top price TT
increase RR%
```

donde `RR` es el porcentaje de dicho incremento (con 2 cifras decimales). Si no existiera ninguna puja, se mostrará el mensaje:

```
Top bid not possible
```

Si la lista de productos estuviese vacía, se imprimirá el mensaje:

```
+ Error: Stats not possible
```

- Si la operación es ***[R]emove***, se eliminarán de la lista aquellos productos que no tengan pujas. Se mostrará el siguiente mensaje:

```
Removing product PP1 seller UU1 category CC1 price ZZ1 bids II1
Removing product PP2 seller UU2 category CC2 price ZZ2 bids II2
Removing product PP3 seller UU3 category CC3 price ZZ3 bids II3
...
Removing product PPn seller UUn category CCn price ZZn bids IIn
```

  Si no existiese ningún producto sin pujas, se mostrará el mensaje:

```
+ Error: Remove not possible
```

4. Lectura de los Ficheros de Entrada

Para facilitar el desarrollo de la práctica se proporciona el siguiente material de especial interés: (1) Un directorio `CLion` que incluye un proyecto plantilla (`P2.zip`) y, (2) Un directorio `script` donde se proporciona un fichero (`script.sh`) que permite probar de manera conjunta los distintos archivos proporcionados. Además, se facilita un documento de ayuda para su ejecución (`EjecucionScript_P2.pdf`). Nótese que, para que el `script` no dé problemas se recomienda **NO copiar directamente el código de este documento**, ya que el formato PDF puede incluir caracteres invisibles que darían por incorrectas salidas (aparentemente) válidas.

5. Información Importante

El documento `NormasEntrega.pdf`, disponible en la página web de la asignatura detalla claramente las normas de entrega. Para un mejor **seguimiento de la práctica** se realizarán dos **entregas parciales obligatorias** antes de las fechas y con los contenidos que se indican a continuación:

- **Entrega parcial #1: Viernes 8 de Abril a las 22:00 horas.** Implementación y prueba del TAD Product List y el TAD Bid Stack (entrega de los ficheros `types.h`, `product_list.c`, `product_list.h`, `bid_stack.c` y `bid_stack.h`).
- **Entrega parcial #2: Viernes 22 de Abril a las 22:00 horas.** Implementación y prueba de las siguientes funcionalidades del programa principal: `New`, `Stats` y `Bid` (entrega de los ficheros `types.h`, `product_list.c`, `product_list.h`, `bid_stack.c`, `bid_stack.h` y `main.c`). Para comprobar el correcto funcionamiento de las operaciones se facilitarán los ficheros de prueba `new.txt` y `bid.txt`.

Se realizará una corrección automática usando el *script* proporcionado para ver si se superan o no los correspondientes requisitos (véase documento `CriteriosEvaluacion.pdf`).

Fecha límite de entrega: **Viernes 29 de abril de 2022 a las 22:00 horas.**